Komunikacja człowiek komputer - sprawozdanie z aplikacji Mask Detector

Gorgoń Adam - 145278
Grochowska Paulina - 145284

9 grudnia 2021

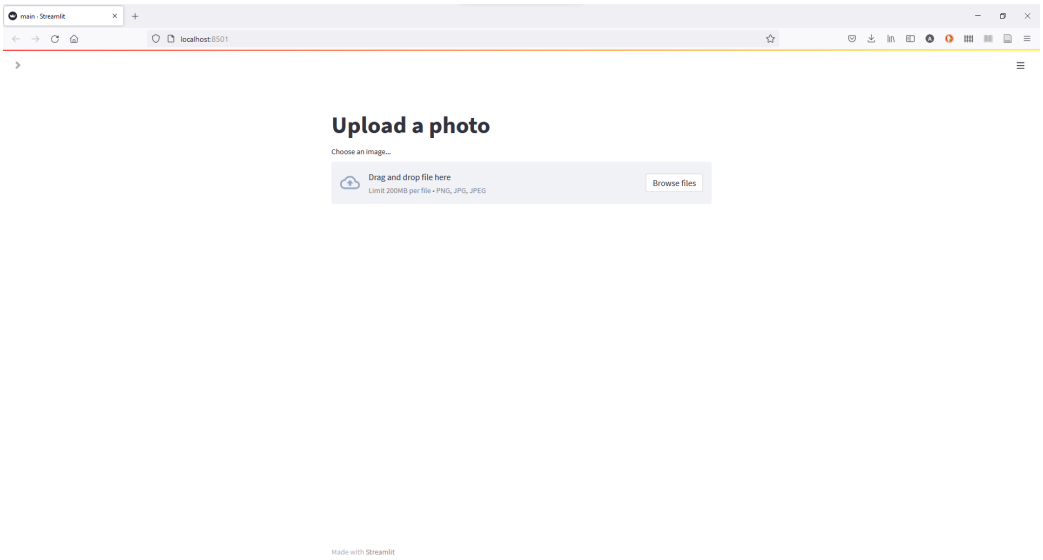
1 Wstęp

Celem projektu jest zaimplementowanie aplikacji do wykrywania, czy osoba ma założoną maskę ochronną. Podczas tworzenia projektu wgłębiliśmy się nie tylko w metody wykrywania twarzy, maski, lecz także w miarę potrzeby poszerzyliśmy swoją wiedzę o strukturze i sposobie działania sieci neuronowej. Nasz program znajduje się na githubie. Dodatkowo trzeba wrzucić pliki z zipa „modele.zip" do folderu data.

2 Prezentacja aplikacji

Po włączeniu aplikacji webowej na ekranie pojawia się przycisk, który umożliwia wgranie wcześniej przygotowanego zdjęcia. Po użyciu komponentu pojawia się okno menedżera plików. Po wybraniu i zatwierdzeniu zdjęcia o formacie .jpg, .jpeg lub .png okno zamyka się, a aplikacja zaczyna przetwarzanie obrazu. Po paru sekundach w przeglądarce pojawia się nasze zdjęcie z modyfikacjami.

Pierwszą z nich jest kolorowa ramka wskazująca na lokalizację twarzy. Może ona wystąpić w dwóch wariantach kolorystycznych: czerwonej lub zielonej. Czerwona oznacza, że po przetworzeniu zdjęcia przez sieć neuronową uznano, iż osoba na zdjęciu nie posiada założonej maseczki ochronnej. W przeciwnym razie, ramka staje się zielona. Dodatkowo nad tym prostokątem można znaleźć informację słowną o posiadaniu ('Mask') lub braku maseczki ('No Mask') w kolorze odpowiadającym kolorze ramki.



Rysunek 1: Aplikacja po otworzeniu

Następną rzeczą są dwie wartości liczbowe oddzielone znakiem '/' zlokalizowane na lewo od napisu. Liczba znajdująca się na lewo od znaku oznacza pewność wykrycia twarzy przez model wykorzystany w algorytmie. Mieści się ona w zakresie $[0, 1]$. Natomiast druga liczba oznacza wartość zwróconą przez sieć neuronową. Dotyczy ona pewności wykrycia maseczki i także mieści się w zakresie $[0, 1]$.

3 Dane

Zbiór danych użytych do trenowania i walidacji modelu pochodzi ze strony Kaggle. Składa się na niego dokładnie 5749 zdjęć, a na każdym z nich znajduje się co najmniej jedna osoba.

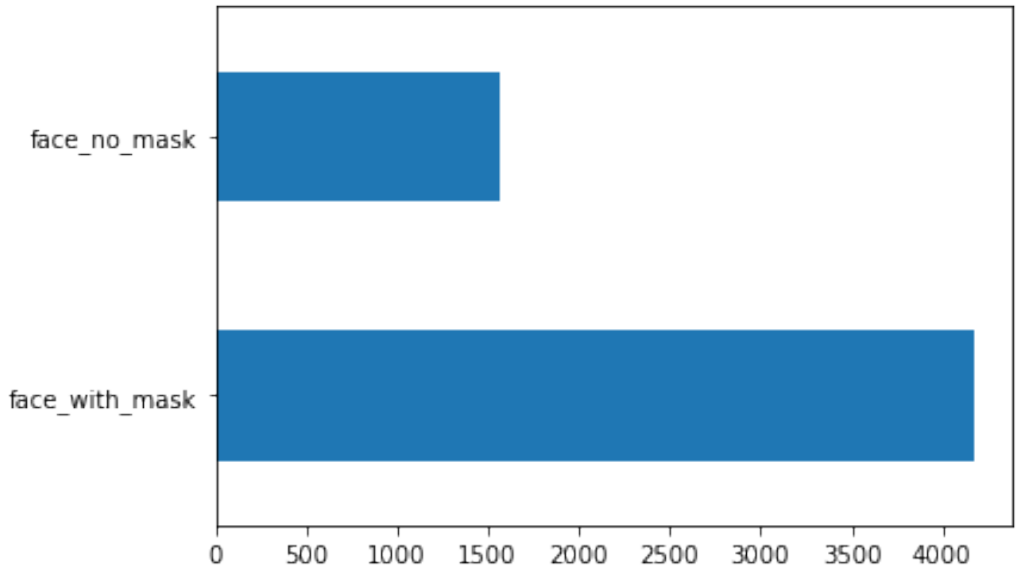
W pliku submission.csv znajduje się tabela z nazwą zdjęcia, koordynatami jednej lub wielu twarzy osoby oraz co najmniej jedną etykietą opisującą osobę/y. Wśród nich:

- 'face_with_mask',
- 'mask_colorful',
- 'face_no_mask',

- 'face_with_mask_incorrect',
- 'mask_surgical',
- 'face_other_covering',
- 'scarf_bandana',
- 'eyeglasses'.

Jednak korzystaliśmy jedynie z dwóch z nich: 'face_with_mask', 'face_no_mask'. W konsekwencji otrzymaliśmy 4180 zdjęć osób w maskach oraz 1569 zdjęć osób bez masek.

Sprawą niezbalansowanych danych zajęliśmy się przy trenowaniu modelu. Dodatkowo do każdego elementu zbioru znajdował się plik .json. Pomógł on w sprawniejszym i dokładniejszym przetworzeniu każdego pliku.



Rysunek 2: Wykres przedstawiający ilość danych

```
{'FileName': '1801.jpg',
        'NumOfAnno': 1,
        'Annotations': [{'isProtected': False,
                'ID': 9248689088688751 36,
```

```
5                   'BoundingBox': [451, 186, 895, 697],
6                   'classname': 'face_no_mask',
7                   'Confidence': 1,
8                   'Attributes': {}}]}
```

Listing 1: Przykład pliku json

4 Użyte modele

Do wykrywania twarzy użyliśmy wytrenowanego modelu z biblioteki OpenCV w frameworku Caffe[2]. Do klasyfikacji, czy osoba posiada maskę stworzyliśmy własną się konwolucyjną zbudowaną na technologi tensorflow[1]. Model trenował się na cpu 3 godziny. Wczytywał zdjęcia z wcześniej wspomnianej bazy. Proporcje zbioru treningowego do testowego, wynosiły 0.8/0.2. Do lepszego wytrenowania model generował sobie zdjęcia, obracając zdjęcia o maksymalnie 15 stopni, robiąc lustrzane odbicie oraz przesuwając w pionie lub poziomie o 0.1 zdjęcia.

5 Działanie programu

5.1 Przygotowanie zdjęcia do modelu

Wczytane zdjęcie jest zmieniane na tablicę jako RBG. Następnie układ jest zmieniany na BGR. Wynika to z faktu, że podczas trenowania dane były wczytywane za pomocą biblioteki OpenCV(wczytuje bgr), a w aplikacji przez Pillow(wczytuje rgb).

Następnie rozjaśniamy obraz za pomocą korekcji gamma, by zmniejszyć efekt cienia na zdjęciu.

5.2 Detekcja twarzy

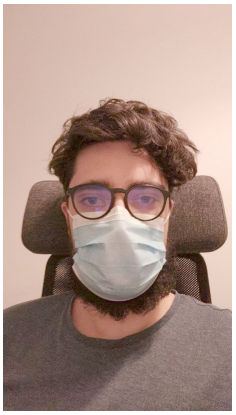
Model sprawdza na zdjęciu czy znajduję się twarz. Jeśli pewność algorytmu na to, że w przeszukiwanym w tym momencie prostokącie znajduje się twarz, wynosi ponad 0.5, to program przechodzi do klasyfikacji.

(a) Zdjęcie bez obróbki z źle wczytanymi kolorami

(b) Zdjęcie bez obróbki z dobrze wczytanymi kolorami

Rysunek 3: Wczytywanie zdjęcia

(a) Przed

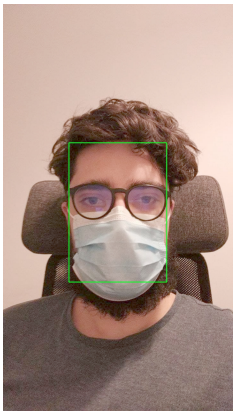


(b) Po

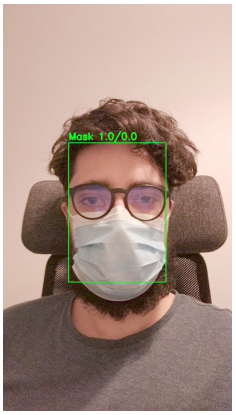
Rysunek 4: Korekcja gamma

5.3 Klasyfikacja twarzy

Jeśli program znajdzie twarz, to "wycina"prostokąt uznany za twarz, zmienia jego rozmiar na 124x124 i wysyła do sieci konwolucyjnej. Sieć wyrzuca liczbę w zakresie $[0, 1]$. Jeśli liczba jest mniejsza od 0.5, to twarz ma na sobie założoną maskę, w przeciwnym razie algorytm uznaje, że na twarzy nie ma maseczki.



(a) Algorytm znalazł twarz



(b) Algorytm sklasyfikował twarz

Rysunek 5: Działanie modelu

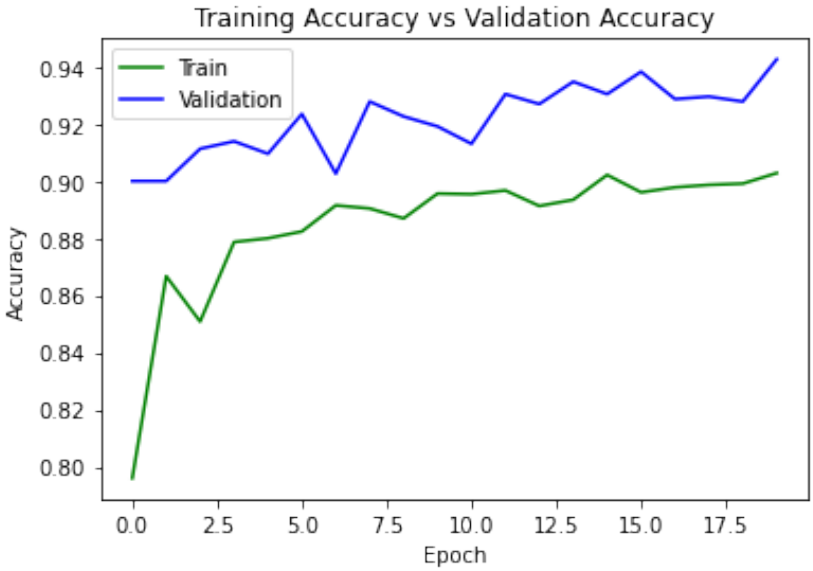
6 Podsumowanie

Po wytrenowaniu modelu i sprawdzeniu działania aplikacji przeanalizowaliśmy otrzymane wyniki i doszliśmy do następujących wniosków. Zdecydowana większość uzyskanych wartości była zgodna z oczekiwanymi (macierz pomyłek, wykres 9). Wystąpienie niepoprawnych dopasowań może wynikać z nieidealnego dobrania parametrów. Niemniej jednak przy zastosowaniu sieci neuronowych obecność błędów jest normalna, a uzyskana pewność jest zadowalająca na skalę, ograniczenia pamięciowe i czasowe aplikacji.

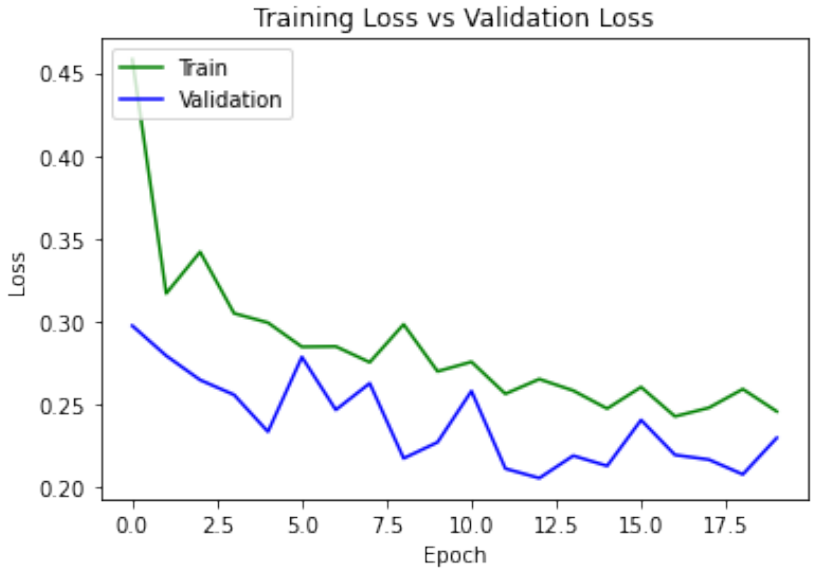
Ponadto patrząc na wykres 8 widać, że dla wartości parametru granicznego równego 0.5 otrzymujemy minimum funkcji. Oznacza to, że parametr o tej wartości zapewnia nam najmniejszą ilość pomyłek, a przy tym najlepsze rezultaty.

Analizując wykresy 6 i 7 można dojść do wniosku, że krzywa odpowiadająca za dane testowe walidacyjne szybko ‘zbliżyła się’ do do drugiej krzywej. Wskazuje

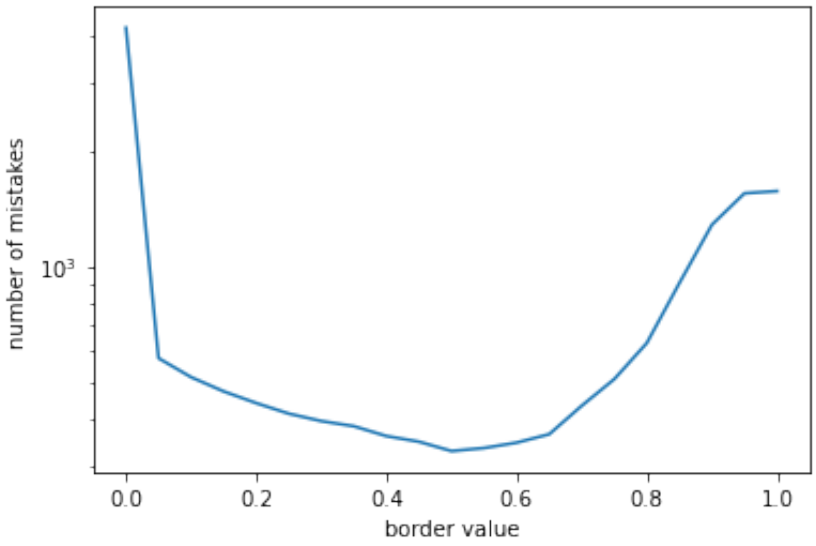
to na szybką poprawę efektywności sieci neuronowej, a także na ogólną dobrą jakość programu.



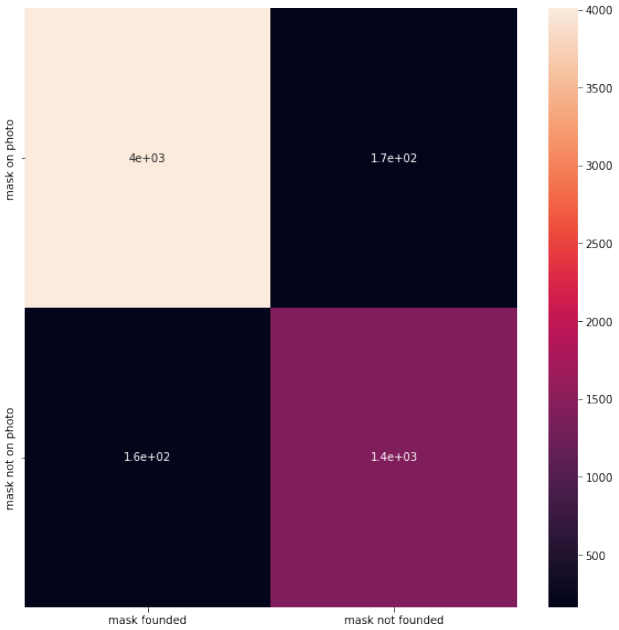
Rysunek 6: Wykres przedstawiający dokładność na danych walidacyjnych i treningowych



Rysunek 7: Wykres przedstawiający straty na danych walidacyjnych i treningowych



Rysunek 8: Wykres przedstawiający ilość pomyłek, przy zmianie parametru granicznego



Rysunek 9: Macierz pomyłek dla parametru granicznego o wartości 0.5

Literatura

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems.

[2] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding.